|  |  |  |  |
| --- | --- | --- | --- |
|  | Python | JS | Php |
| Output | print()  print(a,b)  print(a+b) | console.log()  console.log(a,b)  console.log(a+b) | echo, print(), print\_r()  echo $a,$b  echo $a.$b |
| Input | username = input("Enter username: ") | const readline=require('readline').createInterface({ input: process.stdin, output: process.stdout })  readline.question(`What's your name?`, name => {  console.log(`Hi ${name}!`)  readline.close()  }) | $x=readline(‘what is your answer?’); |
| Exit | quit() | process.exit() | die() |
| Export | Normal saved py | module.exports=testapp; //nodejs default export  module.exports={ testapp }; //nodejs named exp  export default name  export name | Normal saved php files |
| Import | import module //module.py  import module as mx  from module import person1  Usage  module.var module.func() module.Class()  Print(dir(module)) //print all methods on module | const module = require('')  const module, { module, module, ... } = require('')  import name from ''  import name as newname from ''  import {name} from ''  import \* as name from ‘’  Usage  module module.prop module() new module.Class()  name name.prop name() new Name() | require()  require\_once  include()  include\_once()  Usage  Access all vars, funcs, and classes as usual |
| Casting | int(x), str(x), bool(x) | Number(x), String(x), Boolean(x) | (int) |
| Concatenation | a+b | a+b | a.b |
| Array | ["apple", "banana", "cherry"] //list  () //tuple, {} //set | ["apple", "banana", "cherry"] | array("Volvo", "BMW", "Toyota");  ["Volvo", "BMW", "Toyota"] |
| Object | { "brand": "Ford", "model": "Mustang"} //dict | { brand: "Ford", model: "Mustang"} | array("Peter"=>"35", "Ben"=>"37"); |
| Conditions | If (a>b): …elif: …else: | If(a>b){} …else if…else  switch(expression) {  case x: break;  default: } | If(a>b){} …elseif…else  switch(expression) {  case x: break;  default: } |
| Logical | and, or, not, is, is not | &&, ||, ! | and &&, or ||, ! |
| Loops | while i < 6: else:  for x in arr|obj: else: //arr x,obj x obj[x] | for (var i = 0, str=””; i < arr.length; i++) {}  for (let x of arr) {x} //arr  for (let x in obj) {obj[x]} //obj  while (i < 10) { i++ }  do { i++ } while (i < 10); | for ($x = 0; $x <= 10; $x++) {}  foreach ($arr as $x) { } //arr  foreach($obj as $x => $val) { } //obj  while($x <= 5) { $x++ }  do { $x++ } while ($x <= 5); |
| Function | def my\_function(arg=”str”): arg  def my\_function(\*args): args[0],args[1]  x = lambda a,b : a + b  return str, arr, obj  return lambda a: a+10 | function name(arg=”str”){ arg }  var name = function(){ }  var name = () => { }  var x = (x, y) => x \* y;  (function(){ }); (()=>{ })  return str, arr, obj  return function | function name($arg=”str”){ $arg }  $name = function(){ }  (function(){ }); //test this  return str, arr, obj |
| Global | global x | Create var in main body | $GLOBALS[‘x’] |
|  | try:  except:  else: | finally:    raise Exception(‘’) | try {}  catch(err) {}  finally {}  throw 'err'; | try {}  catch (\Throwable $th) { //throw $th;}  finally{}  throw 'err'; |
| Class | class Person:  def \_\_init\_\_(this, name, age):  this.name = name  this.age = age  x=5  def myfunc(this):  print("Hello my name is " + this.name)  p1 = Person("John", 36)  print(p1.x)  print(p1.name)  print(p1.age)  p1.myfunc()  p1.age = 40  p1.x=15  //self can replace this, alwasy 1st arg in all funcs | class Person {  constructor( name, age){  this.name = name  this.age = age }  var x=5  static hello() { console.log( "Hello!!") }  myfunc(){ console.log("Hello my name is " + this.name) }  }  var p1 = new Person("John", 36)  console.log (p1.x)  console.log (p1.name)  console.log (p1.age)  p1.myfunc()  Person.hello()  p1.age = 40  p1.x=15  function person(name,age) {  this.name=str;  this.age=age;  var x= 5;  function hello(){ console.log('Hello')}  this.myfunc=function(){ console.log("Hello my name is " + this.name)}  return this.name+this.age;  }  person.prototype.new = "08976567";  person.prototype.age = function() {console.log("Hello my age is " + this.age) };  var p1= new person(("John", 36)  console.log (p1.x)  console.log (p1.name)  console.log (p1.age)  p1.myfunc()  person().hello()  person() | trait message {  public function msg() { echo "OOP is fun! ";}  }  class Person {  use message;  public $name, $age;  public function \_\_construct($name,$age) {  $this->name = $name;  $this->age = $age;  }  public $x=5; protected $y; private $z;  static $sch="new school";  static function hello() { echo "Hello!!"; }  function myfunc() { echo "Hello my name is ".$this->name; }  function \_\_destruct() { echo "The class is over {$this->name}"; }  //access any var,func with self::$value|$func(); or $this->value|func()  // use final on class or method to sto override  }  $p1 = new Person("John", 36);  $p1->msg();  $p1->myfunc();  echo $p1->name;  echo $p1->age;  echo $p1->x;  echo Person::$sch;  Person::hello();  $p1->age=30;  $p1->x=34; |
| Class Inheritance | class Student(Person): //inherits Person  def \_\_init\_\_(this, name, age, class):  super().\_\_init\_\_(name, age)  this.class=class  def welcome(this):  print("This " + this.name + " "+ this.age, ":the class of", this.class )  x = Student("Mike", 40, 2019) | class Student extends Person{ //inherits Person  constructor(name, age, class){  super(name,age)  this.class=class }  welcome(){  console.log("This " + this.name + " "+ this.age, ":the class of", this.class) }  }  var x = new Student("Mike", 40, 2019) | class Student extends Person{ //inherits Person  function \_\_construct($class){  $this->class=$class; }  function welcome(){  echo "This " + $this->name + " "+ $this->age, ":the class of", $this->class; }  }  $x = new Student("Mike", 40, 2019); |
| Accept CLI Arg |  |  |  |